The *TeraPaths* API

1. Authentication/Authorization

The TeraPaths API uses mutual authentication with X.509 certificates.
Currently, the certificates are DOE issued. A java client to the API needs to have a keystore and a truststore setup properly and include code such as the following:

```
// Begin SSL security stuff...
try {
    System.setProperty("javax.net.ssl.keyStoreType", "JKS");
    System.setProperty("javax.net.ssl.keyStore", keyStoreDir + "terapathsClientKeystore.jks");
    System.setProperty("javax.net.ssl.keyStorePassword", keyStorePassword);
    System.setProperty("javax.net.ssl.trustStoreType", "JKS");
    System.setProperty("javax.net.ssl.trustStore", keyStoreDir + "terapathsClientTruststore.jks");
    System.setProperty("javax.net.ssl.trustStorePassword", trustStorePassword);
 } catch (Exception e) {
    System.out.println("Failed to set SSL properties...");
    e.printStackTrace();
 }
 // End SSL security stuff
```

The keystore contains the user's certificate chain. The private key password needs to be the same as the keystore password.
The truststore contains the public keys of the TeraPaths servers and the DOE CA.
Additionally, a user's DN needs to be included in the TeraPaths VO and assigned a suitable role.

2. Basic methods

tpsAPI_reserve

ReservationData tpsAPI_reserve(ReservationData request)
Creates a set of temporary resource reservations for a virtual path between the source and destination with the specified QoS and bandwidth parameters. Depending on system configuration, this set may involve reservations in some or all domains along the path between source and destination. The reservation set has to be committed within 60 seconds otherwise the resources are released.
Parameters:
A ReservationData object with information regarding the start time and duration of the reservation, the bandwidth and QoS class, source and destination addresses and port numbers.
Returns:
A ReservationData object with the details of the successfully reserved reservations or null if the reservation failed. If modification is allowed and a suitable slot is found the start time will be adjusted accordingly.
Throws:
java.rmi.RemoteException

tpsAPI_commit

boolean tpsAPI_commit(ReservationData rd)
Commits a temporary reservation set after a reserve call. The commit call confirms the reservation of resources allocated temporarily during the reserve phase.
Parameters:
A ReservationData object with the id field set to the id of the reservation to be commited.
Returns:
true if the reservation is committed successfully, ***false*** otherwise.
Throws:
java.rmi.RemoteException

tpsAPI_cancel

boolean tpsAPI_cancel(ReservationData rd)
Cancels a reservation set with the reservation id specified in the reservationData input.
Parameters:
A ReservationData object with the id field set to the id of the reservation to be cancelled.
Returns:
true if the reservation was cancelled successfully, ***false*** otherwise.
Throws:
java.rmi.RemoteException

3. Auxiliary methods

tpsAPI_getBandwidths

Bandwidths[] tpsAPI_getBandwidths(String srcIp, String destIp)
Returns the name and maximum reservable bandwidth of all defined traffic classes available at the specified source and destination.
Parameters:
String srcIp – source IP or CIDR address/address list
String destIp – destination IP or CIDR address/address list
Returns:
Names and maximum reservable bandwidths of traffic classes in a Bandwidths object.
Bandwidths[0] Information about local bandwidth classes.
Bandwidths[1] left empty by convention
Bandwidths[2] Information about remote bandwidth classes.
Throws:
java.rmi.RemoteException

tpsAPI_getPath

String[] tpsAPI_getPath(String srcIp, String destIp)
Returns the URLs of network controllers that must be contacted to setup a virtual path between the specified source and the destination.
Parameters:
String srcIp – source IP or CIDR address/address list
String destIp – destination IP or CIDR address/address list
Returns:
A String array containing three URLs:
String[0] local TeraPaths API URL
String[1] WAN controller URL
String[2] remote TeraPaths API URL
Throws:
java.rmi.RemoteException

4. Privileged methods (system communication)

tpsAPI_LocalCancel

boolean tpsAPI_LocalCancel(String id, String userDN, String userCA) throws java.rmi.RemoteException
The Function can only be invoked by privileged users (remote TeraPaths Instance or Network Administrators) to cancel a reservation (local domain only).
Parameters:
id – the id of the reservation to be locally cancelled.
userDN – the DN of the user that initiated this cancel.
userCA – the CA that issued the users certificate.
Returns:
true if the reservation was successfully cancelled locally, ***false*** otherwise.
Throws:
java.rmi.RemoteException

tpsAPI_LocalCommit

boolean tpsAPI_LocalCommit(String id, String userDN, String userCA) throws java.rmi.RemoteException
The function can only be invoked by privileged users (remote TeraPaths Instance or Network Administrators) to commit a reservation (local domain only).
Parameters:
id – the id of the reservation to be locally committed.
userDN – the DN of the user that initiated this commit.
userCA – the CA that issued the users certificate.
Returns:
true if the reservation was successfully committed locally, ***false*** otherwise.
Throws:
java.rmi.RemoteException

tpsAPI_LocalRemove

boolean tpsAPI_LocalRemove(String id, String userDN, String userCA) throws java.rmi.RemoteException
Invoked by privileged users (remote TeraPaths Instance or Network Administrators) this function is used to remove a reservation from the database (local domain only).
Parameters:
id – the id of the reservation to be locally removed.
userDN – the DN of the user that initiated this remove.
userCA – the CA that issued the users certificate.
Returns:
true if the reservation was successfully removed locally, ***false*** otherwise.
Throws:
java.rmi.RemoteException

tpsAPI_LocalReserve

ReservationData tpsAPI_LocalReserve(ReservationData rd) throws java.rmi.RemoteException
Invoked by privileged users (remote TeraPaths Instance or Network Administrators) this function creates a temporary reservation for a virtual path across the local domain.
Parameters:
A reservationData object set to all the appropriate reservation specifications such as start time, duration, QoS class, bandwidth, source and destination IP/CIDR addresses/lists and port numbers.
Returns:
A reservationData object with assigned reservation id (if not already assigned), and adjusted start time and duration (if modification is allowed, necessary, and possible).
Throws:
java.rmi.RemoteException

tpsAPI_LocalStart

boolean tpsAPI_LocalStart(String id, String userDN, String userCA) throws java.rmi.RemoteException
Invoked by privileged users (remote TeraPaths Instance or Network Administrators) this method is the final step in submitting a reservation. It initializes the activation and deactivation tasks.
Parameters:
id – the id of the reservation to be locally started.
userDN – the DN of the user that initiated this remove.
userCA – the CA that issued the users certificate.
Returns:
true if the reservation was successfully started, ***false*** otherwise.
Throws:
throws java.rmi.RemoteException

tpsAPI_getLocalBandwidths

Bandwidth[] tpsAPI_getLocalBandwidths() throws java.rmi.RemoteException
Invoked by privileged users (remote TeraPaths Instance or Network Administrators) this method returns details of the local reservation bandwidth classes.
Parameters:
None
Returns:
An array of Banwidth objects, which hold the details for each of the bandwidth class supported by the local site.
Throws:
java.rmi.RemoteException

tpsAPI_addRelatedReservationId

boolean tpsAPI_addRelatedReservationId(String rid, String rrid, boolean replace) throws java.rmi.RemoteException
Invoked by privileged users (remote TeraPaths Instance or Network Administrators) this function updates the relatedReservation entry for the specified reservation in the table.
Parameters:
String rid – the id of the reservation whose related reservations entry is to be modified.
String rrid – the id of the related reservation to be added to or replace the entry in the database.
boolean replace – flag set when you want to overwrite the existing entry.
Returns:
true if the database was successfully updated, ***false*** otherwise.
Throws:
java.rmi.RemoteException

tpsAPI_getRelatedReservationIds

String tpsAPI_getRelatedReservationIds(String id) throws java.rmi.RemoteException
Invoked by privileged users (remote TeraPaths Instance or Network Administrators) this function provides information regarding the WAN reservations associated with the reservation identified by the id.
Parameters:
id – the id of the reservation whose related reservation ids are required.
Returns:
A comma-delimited string containing the list of related reservations.
Throws:
java.rmi.RemoteException

tpsAPI_getReservationData

ReservationData tpsAPI_getReservationData(String id) throws java.rmi.RemoteException
Invoked by privileged users (remote TeraPaths Instance or Network Administrators) or the owner of this reservation to get the details of the reservation with the corresponding id.
Parameters:
id – the id of the reservation whose information is required.
Returns:
The details of the reservation identified by the reservation id in a reservationData object.
Throws:
java.rmi.RemoteException

tpsAPI_lookupUser

UserData tpsAPI_lookupUser(String userDN, String userCA) throws java.rmi.RemoteException
Invoked by privileged users (remote TeraPaths Instance or Network Administrators) to lookup user data in the local database.
Parameters:
userDN – the DN of the user.
userCA – the CA that issued the users certificate.
Returns:
A userData object with the credentials of the user.
Throws:
java.rmi.RemoteException

getAllReservationsForClass

ReservationData[] getAllReservationsForClass(ReservationData rd) throws java.rmi.RemoteException
Invoked by privileged users (remote TeraPaths Instance or Network Administrators) to retrieve reservation data from the local database. This method is intended to be used by the TeraPaths web interface.
Parameters:
rd – a reservationData object is used to pass search parameters.
Returns:
An array of reservationData objects containing information on reservations existing in the local database.
Throws:
java.rmi.RemoteException

DATA TYPES

Class ReservationData

Holds all the information about a particular reservation

class **ReservationData**
{
```
   private tpsLib.Who who;
   private String userName;
   private String protocol;
   private String srcIp;
   private String srcPorts;
   private String srcPortMin;
   private String srcPortMax;
   private String srcMapping;
   private String destIp;
   private String destPorts;
   private String destPortMin;
   private String destPortMax;
   private String destMapping;
   private String Mapping;
   private tpsLib.Bandwidth bandwidth;
   private long startTime;
   private long dTMinus;
   private long dTPlus;
   private int modifyReservation;
   private long startTimeMin;
   private long startTimeMax;
   private long duration;
   private String id;
   private String srcName;
   private String destName;
   private long timeout;
   private String status;
   private String direction;
   private String relatedReservationIds;

  ...
```
}

Field	Type	Description	Value	Comments
Who	Who	user identity		internal; see class Who
username	String	user name		input; obsolete
protocol	String	communication protocol list	tcp/udp	input
srcIp	String	source IP address list	aaa.bbb.ccc.ddd/xx (CIDR block)	input
srcPorts	String	Source ports list	x/y-z (≥0 ≤ 65535)	input
srcPortMin	int	source port range start		input; obsolete
srcPortMax	int	source port range end		input; obsolete
srcMapping	String	source mapping mode (ip-ports)	strict/combination	input
destIp	String	destination IP address	aaa.bbb.ccc.ddd/xx (CIDR block)	input
dstPorts	String	destination ports list	x/y-z (≥0 ≤ 65535)	input
destPortMin	int	destination port range start		input; obsolete
destPortMax	int	destination port range end		input; obsolete
destMapping	String	destination mapping mode (ip-ports)	strict/combination	input
Mapping	String	source-destination mapping mode	strict/combination	input
bandwidth	Bandwidth	QoS service class		input; see class Bandwidth
startTime	long	reservation start time	epoch time (milliseconds)	input
dTMinus	long	tolerance period before startTime		experimental use (must set to 0)
dTPlus	long	tolerance period after startTime		experimental use (must set to 0)
modifyReservation	int	reservation modification on/off	0/1	input
startTimeMin	long	earliest acceptable startTime	epoch time (milliseconds)	input; optional
startTimeMax	long	latest acceptable startTime	epoch time (milliseconds)	input; optional
duration	long	reservation duration in seconds	seconds	input
id	String	reservation identity tag		internal use; input (commit/cancel)
srcName	String	source host name		internal use
destName	String	destination host name		internal use
timeout	long	reservation end time		internal use
status	String	reservation status		internal use
direction	String	reservation direction	unidirectional/bidirectional	input
relatedReservationIds	String	related WAN reservation tags		internal use

The mapping fields affect the correspondence of addresses to ports and sources to destinations. ***strict*** correspondence is one-on-one, ***combination*** will make the system figure out all possible combinations.
For example:
x: s or d for source or destination
n = 1,2,…
ixn denotes a CIDR address block
pxn denotes a port or a "-" delimited port range
(ixn,pxn) denotes a flow group source or destination (fs or fd)
[fs,fd] denotes a flow group (f/fg)
{fg,fg,…} denotes the set of flow groups affected by the reservation

srcIp=is1, is2
srcPorts=ps1,ps2
srcMapping=strict
destIp=id1,id2
destPorts=pd1,pd2
destMapping=strict
Mapping=strict

resolves to {[(is1,ps1),(id1,pd1)],[(is2,ps2),(id2,pd2)]}

srcIp=is1, is2
srcPorts=ps1,ps2
srcMapping=strict
destIp=id1,id2
destPorts=pd1,pd2
destMapping=strict
Mapping=combination

resolves to {[(is1,ps1),(id1,pd1)],[(is1,ps1),(id2,pd2)],[(is2,ps2),(id1,pd1)],[(is2,ps2),(id2,pd2)]}

Setting all mappings to ***combination*** in the above example results in a set with 16 flow groups
Sn error occurs if the elements are not enough for a strict match.

Class Bandwidth

QoS class name and bandwidth to allocate

```
public class Bandwidth
{
   private String className;
   private long bandwidth;

}
```

Field	Type	Description	Value	Comments
className	String	QoS class name	BE/EF (others possible)	input
bandwidth	long	bandwidth	bits per second	input

Class Who

Holds user credentials from X.509 certificate. The credentials are extracted from the SSL exchange and filled in automatically.

```
public class Who
{
   private String name;
   private String DN;
   private String CA;

   …

}
```

Field	Type	Description	Value	Comments
Name	String	QoS class name		obsolete
DN	String	Certificate DN		internal use
CA	String	Certificate CA		internal use